# ■ Missing Number – DSA Notes

## ■ Problem Statement

Given an array of length **n** containing distinct numbers in range 0 to n, find the missing number using an algorithm with **linear time complexity (O(n))**.

## ■ Examples

**Example 1:**
Input: n = 3, nums = [3, 0, 1]
Output: 2
Explanation: Range {0,1,2,3} → 2 is missing.

**Example 2:**
Input: n = 4, nums = [1, 2, 3, 4]
Output: 0
Explanation: Range {0,1,2,3,4} → 0 is missing.

**Example 3:**
Input: n = 4, nums = [0,1,2,3]
Output: 4
Explanation: Expected range {0,1,2,3,4} → 4 is missing.

## ■ Core Logic (Math Formula)

1. The sum of first n natural numbers = **n × (n + 1) / 2**.
2. Compute total sum of array elements.
3. Missing number = **expected_sum - actual_sum**.
4. Works because values are distinct and range is fixed.

## ■ Dry Run (nums = [3,0,1])

| n | Expected Sum (n(n+1)/2) | Actual Sum | Missing Number |
|---|---|---|---|
| 3 | 3×4/2 = 6 | 3 + 0 + 1 = 4 | 6 - 4 = 2 |

## ■ C++ Code

```
class Solution {
  public:
  int missingNumber(vector &nums;) {
    int n = nums.size();
    int total = n * (n + 1) / 2;
    int sum = 0;

    for (int i = 0; i < n; i++) {
      sum += nums[i];
    }
    return total - sum;
  }
};
```

## ■■ Time & Space Complexity

**Time Complexity:** O(n) — single loop
**Space Complexity:** O(1) — no extra storage

## ■ Key Notes

- Uses mathematical formula; fastest method.
- No sorting required.
- Works for any valid permutation of 0…n.